17/07/2012

EXPRESSÕES REGULARES

Rodrigo M Mar

"Você preencheu os dados de forma errada! Preencha novamente, volte para o final da fila, espere mais meia hora para ser atendido e reze para tudo dar certo..."

Essa realidade, muito comum quando tentamos obter algum serviço, tira qualquer um do sério, correto? O equivalente online também existe: você está navegando na Internet, preenche algum formulário de cadastro ou de acesso ao serviço online que acabou de descobrir e recebe um aviso de que os dados são inválidos e que deve tentar novamente. Ninguém merece isso...

A verificação das informações que um usuário fornece é parte importante de qualquer aplicativo online. Os motivos são vários. De um lado, um bom aplicativo deve dar ao usuário uma experiência fácil, mostrando se o que ele digita é aceito ou não. Assim, evita-se que o usuário perca seu tempo enviando os dados para saber, alguns segundos depois, que há algo errado com eles. De outro lado, o aplicativo precisa garantir que informação válida seja enviada para processamento e, por segurança, tem que impedir que conteúdo inválido, suspeito ou perigoso seja fornecido.

A validação pode ser feita comparando os dados que o usuário fornece com algum padrão identificado como bom para o aplicativo ou página onde ele está navegando. Neste artigo, vamos conhecer uma forma muito utilizada para criar regras de validação de dados: as expressões regulares. A principio vamos entender a sintaxe das expressões regulares. E então vamos ver alguns exemplos para compreendê-las e perceber como são uma ferramenta poderosa para o programador.

O que são expressões regulares?

Uma expressão regular é um padrão que identifica uma porção de texto. Elas são utilizadas para criar regras com o objetivo de validar um dado, por exemplo, verificar se o que foi digitado é um

endereço válido de e-mail, se é um CPF ou CNPJ, se uma senha contém apenas letras ou se combina letras e caracteres especiais, etc.

A idéia da validação é simples: (a) construa uma regra (a expressão regular) e (b) verifique se os dados digitados se "encaixam" na regra.

Uma regra é composta por caracteres especiais que representam o comportamento ou tipo de dado. Alguns exemplos de regras que podemos construir para identificar o dado são: "possui @", "contém apenas números", "é composto por 11 números", "deve conter pelo menos 1 ponto", entre outros. Esses caracteres especiais, que serão usados para construir nossas regras, são chamados de *metacaracteres*.

Embora a forma de construir as expressões regulares possam variar, dependendo da linguagem de programação utilizada, podemos citar alguns metacaracteres básicos de uma expressão:

. ? * + ^ $ | [ ] { } ( )

Vamos ver alguns exemplos básicos do uso destes caracteres. Ao final do post, apresentaremos uma tabela com uma explicação do comportamento identificado por cada um destes metacaracteres.

Exemplos de Uso:

Vamos validar alguns campos utilizando expressões regulares em JavaScript.

- **CPF**

  Para simplificar o exemplo, não vamos levar em conta que os 2 dígitos ao final de um CPF são dígitos de verificação, calculados a partir dos 9 primeiros números. Vamos validar o formato do CPF, partindo de alguns pressupostos muito simples (e incompletos): o CPF é composto apenas por números e, opcionalmente pode ter 2 caracteres especiais: o ponto e o hifen. Isso nos leva a dois padrões que podem ser identificados:

  111.222.333-44 (números com pontos e um hifen) ou
  11122233344 (apenas números)

  Para simplificar a expressão, vamos considerar qualquer outro formato diferente destes apresentados acima como resultados inválidos.

  A expressão que "encaixa" ou "valida"[1] nosso CPF é:

```
/^[0-9]{3}.?[0-9]{3}.?[0-9]{3}-?[0-9]{2}/
```

Vamos explicar esta expressão:

A barra **/** indica o inicio da nossa regra, o **^** (acento circunflexo) indica a obrigatoriedade de iniciar os dados apenas com **[0-9]** números e é importante que sejam **{3}** números.

O ponto tem um significado especial nas expressões regulares, mas no caso do CPF, queremos que o ponto signifique apenas um ponto. Para isso, usamos a barra invertida que é conhecida como um caractere de escape e diz que o caractere que vem na frente deve ser interpretado como um caractere comum e não um metacaractere. Ou seja, o ponto a seguir é parte do CPF. Este ponto é opcional. Isso é indicado pela interrogação seguida do ponto: **.?** , o mesmo serve para o hifen no final da expressão que também é opcional: **-?** .

No final da regra, **[0-9]{2}** indica que esperamos encontrar apenas números e que eles devem ser exatamente duas repetições.

Vamos testar nossa expressão interativamente? Digite um número válido de CPF abaixo e clique o botão para ver se nossa expressão consegue perceber se ele é válido. Depois faça o mesmo com um número inválido, ou um uma palavra:

CPF Validar

Nesta primeira situação, a validação do CPF é muito simples, apenas para o formato xxx.xxx.xxx-xx (números com pontos e traço). Se você digitar um CPF mal formatado, contendo apenas um ponto, ex: 111.222333-44, o teste acima ainda retornará uma mensagem de CPF válido.

Para evitar esse tipo de erro, vamos alterar nossa regra, que agora fica um pouco mais complexa:

```
/^(([0-9]{3}.[0-9]{3}.[0-9]{3}-[0-9]{2})|([0-9]{11}))$/
```

A barra vertical **|** representa o operador lógico OU. Na expressão acima estamos dizendo que podemos ter um CPF contendo **números, pontos e hifen ou** um CPF **contendo apenas números**, ou seja, algo assim:
111.222.333-44 ou 11122233344

Experimente validar o número de CPF nos formatos acima:

CPF Validar

Enfim, você pode observar que não é tão complexo quanto parece, com alguns exercícios você dominará as expressões regulares a ponto de resolver problemas de uma complexidade muito maior. A tabela abaixo vai ajudá-lo a compreender o significado de alguns metacaracteres.

Guia rápido tabela de metacaracteres

Tabela de Metacaracteres

Metacarcter	Descrição
/ /	Indica nossa expressão regular Exemplo
^	Indica uma pesquisa somente no inicio do bloco/regra. Exemplo
$	Indica uma pesquisa somente no fim do bloco/regra. Exemplo
[ ]	Definir uma classe/lista de caracteres. Exemplo
[^]	Utilizado para negar uma classe/lista de caracteres. Exemplo
( )	Defini um bloco/clausula de caracteres. Exemplo
{x}	Repetir exatamente x de vezes. Exemplo
{x,y}	Repetir no minimo x ou no máximo y de vezes. Exemplo
?	Declara um ou mais caracteres como opcional. Exemplo
+	Repete pelo menos uma vez. Exemplo
|	Representa o conector logico (ou). Exemplo
\	Utilizado para representar um caractere literal. Exemplo
.	Indica qualquer caractere. Exemplo
*	Indica nenhuma ou varias ocorrências de um ou vários caracteres numa regra.

[1] O termo equivalente em inglês é "*match*". Dizemos que uma expressão se encaixa em um padrão: "*an expression matches a pattern*".

Tags: java script, formulários

Add a comment...

Post to Facebook Posting as Jl Vaz (Not you?) Comment

Adriano Antonio da Silva · UNIP - Universidade Paulista

Excelente artigo sobre expressões regulares.

Reply · Like · Follow Post · June 20 at 10:36am

Facebook social plugin

Tweets Follow @sitesesistemas

Opera House @sitesesistemas 27 Sep

A Pen by Opera House codepen.io/operahouse/det... via @codepen

Show Summary

Tweet to @sitesesistemas

Links

Criação de Sites

Sistemas Online

Loja Virtual e Catálogo de Produtos

Criação de Logotipo e Material Gráfico

Desenvolvido por

OPERA HOUSE